Performance of a lattice Boltzmann solver Musubi on various HPC systems

J. Qi, K. Jain, H. Klimach, S. Roller

Chair of Simulation Techniques & Scientific Computing

University of Siegen, Germany

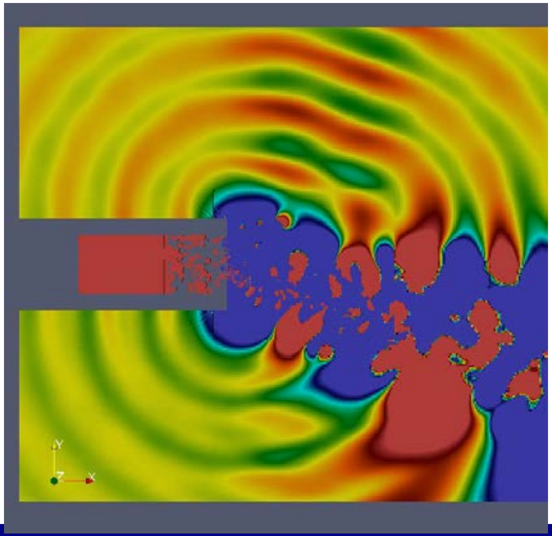
Contents

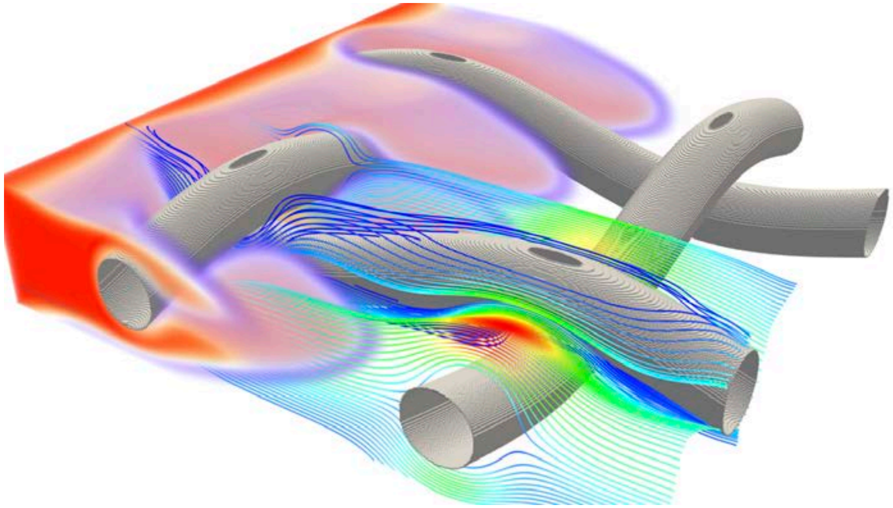
Keywords: HPC; parallel computing; octree; lattice Boltzmann method

Motivation

Flow simulation in arbitrarily complex geometry, which might also involves multiple physical phenomena and scales.

– Aero-acoustic, multi-species, hemodynamics





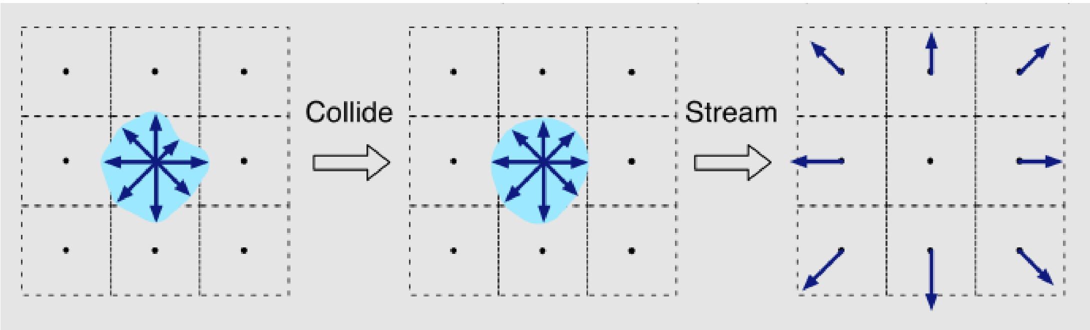
Contents

- Motivation
- **Lattice Boltzmann Method**
- Our code - Musubi
  - Octree data structure
  - implementation
- Performance on various HPC systems
  - Juqueen (IBM BlueGene/Q)
  - Kabuki (NEC SX-ACE)
  - Hornet (Cray XC40)
  - SuperMUC (IBM BlueGene/Q)
- Summary

Keywords: HPC; parallel computing; octree; lattice Boltzmann method

Lattice Boltzmann Method

- It simulates incompressible fluid flows at kinetic level.
  - Cartesian uniform grids.
  - Each element has 19 degree of freedom (DoF) in 3D.
- Iteration based algorithm
  - Collision: purely local computation
  - Streaming: data moving between nearest neighbor (gathering or scattering)



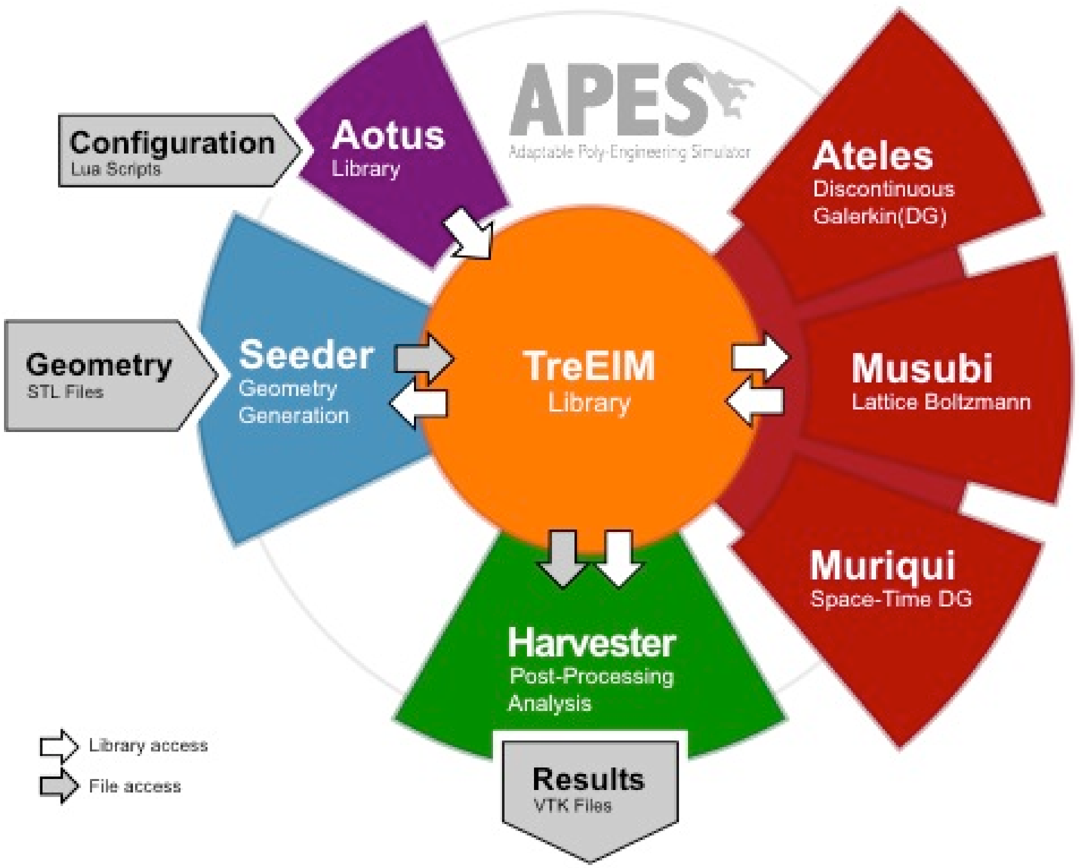
Why to choose LBM

- Simple algorithm
- Parallelization
- Complex geometry
- Multi-physics, multi-species
- but it also has the other side:
  - LBM is still young
  - low mach number flow
  - high resolution

Program structure:

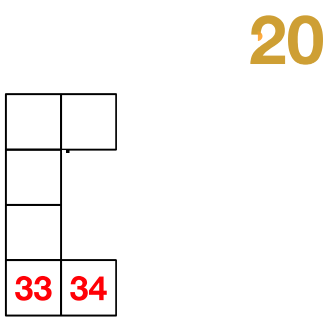
```
Init
do iStep = 1, nSteps
        call BC
        call kernel
                do iElem = 1, nElems
                        compute…
                end do
        call communication
        call others
end do
Finalize
```

APES: End-to-End simulation toolkit



Linearized octree

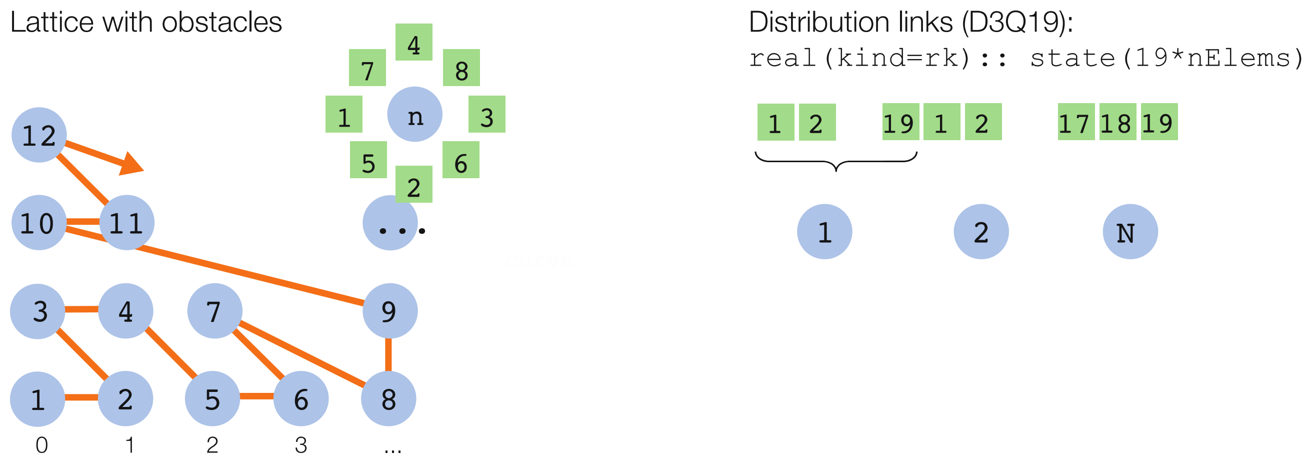
- Every elements are numbered following a *space filling curve*.
- This unique number is called *treeID*, saved as a 8 bytes integer. With another 8 bytes saving property, each element requires 16 bytes only.
- Only leaves are stored, equally distributed among all processes.



Why to use octree?

- the best compromise between easily distributed structured meshes and highly flexible unstructured meshes.
- keep the global information on all processes at a minimum
- efficient neighbor identification in distributed computations
  - treeID implicitly contains geometrical and topological information
- Small size of mesh file, parallel I/O
  - 1 million elements = 16 MB, 1 billion = 16 GB
- Good locality maintained by space filling curve.
  - Important for cache-based CPUs.

Optimization



- Sparse matrix storage: only fluid elements are stored.
- Locality: elements arranged according to space-filling curve.
- Indirect addressing: data stored in 1D array with connectivity list.
- Combined stream-collision.
- Implicit solid boundary treatment.

implementation

- Musubi is written in mostly Fortran 2003, minor in C (Lua lib .etc)
- PULL scheme: streaming (gathering), collision (bgk)
- state variables in double buffers
  - switch after each iteration
- array-of-structure (AOS) data layout
  - i.e. f(1:19,1:M,0:1)
- B_{code} (Bytes/FLOPS) = 3.325 > $B_{machine}$
  - mostly $B_{machine} < 1$
  - Memory bandwidth is the limitation

Benchmark analysis

- fully periodic cubic geometry without any boundary conditions
- Absolute performance
  - *MLUPS* (= million lattice upda
- Performance in parallel:
  - *MLUPSpN* = *MLUPS* per node
- intra-node, inter-node scena

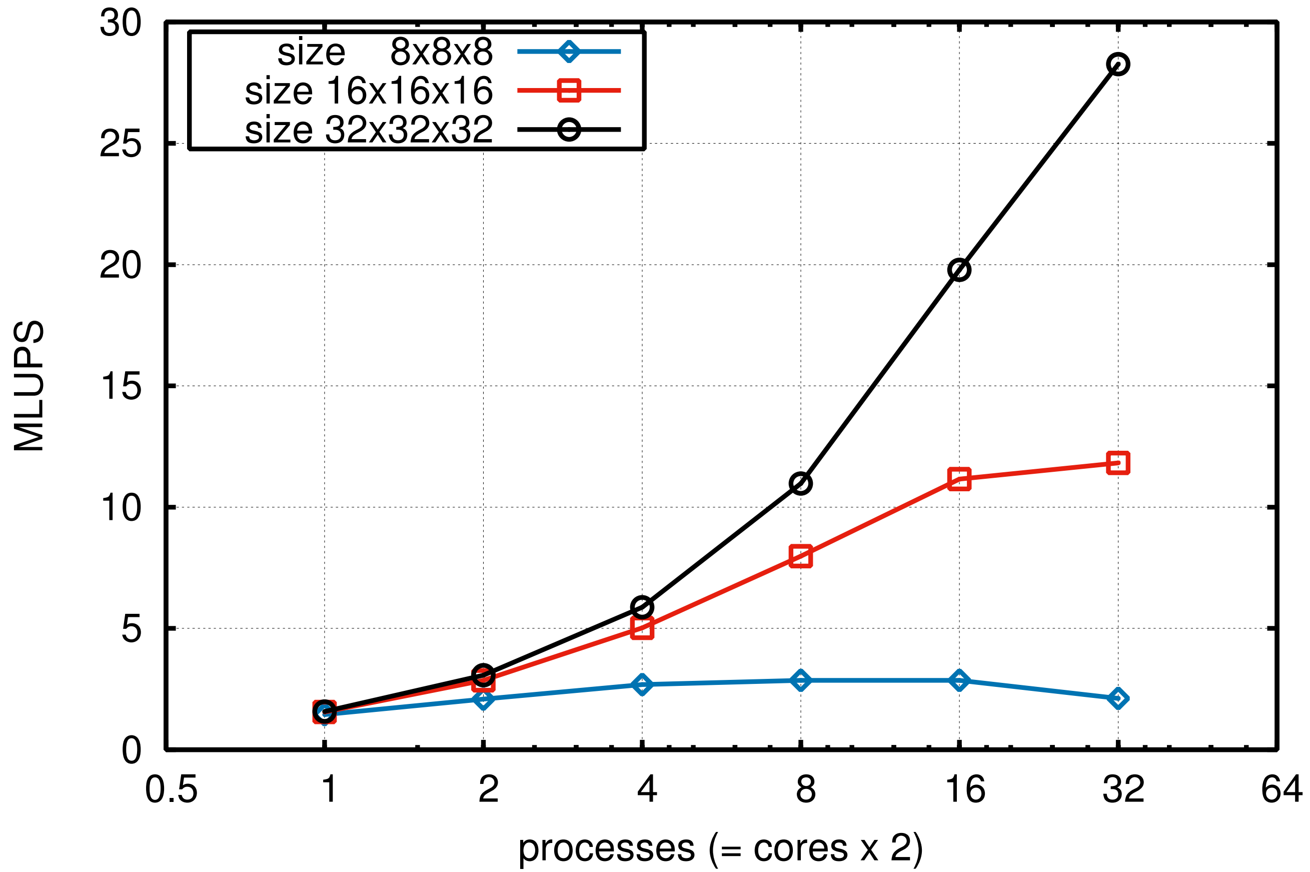
Program structure:

```
Init
do iStep = 1, nSteps
        call BC
        call kernel
                do iElem = 1, nElems
                        compute…
                end do
        call communication
        call others
end do
Finalize
```

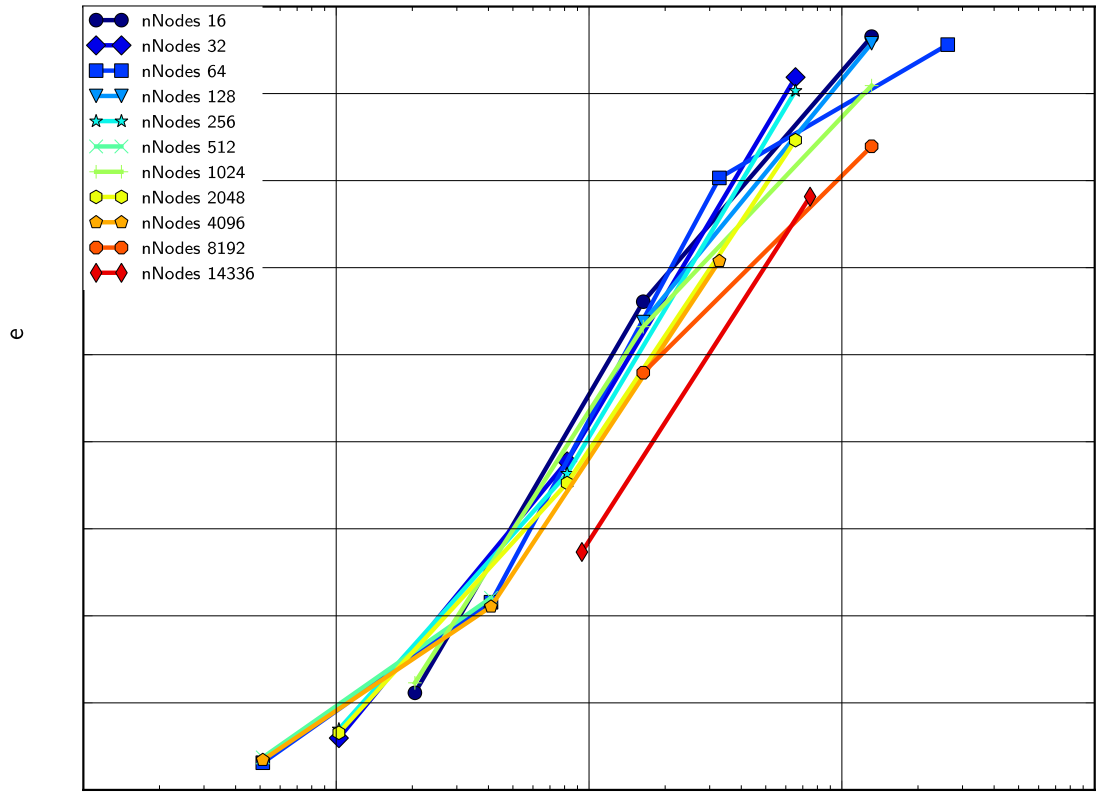
Juqueen

- IBM BlueGene/Q with PowerPC A2
- 16 cores @1.6 GHz with 16 GB memory
  - each core with 2 threads
- compiled using IBM XL Fortran 12.1 with -O3
- Only tested relatively small problem size
  - limit memory
  - initialization (recursive) took too long time

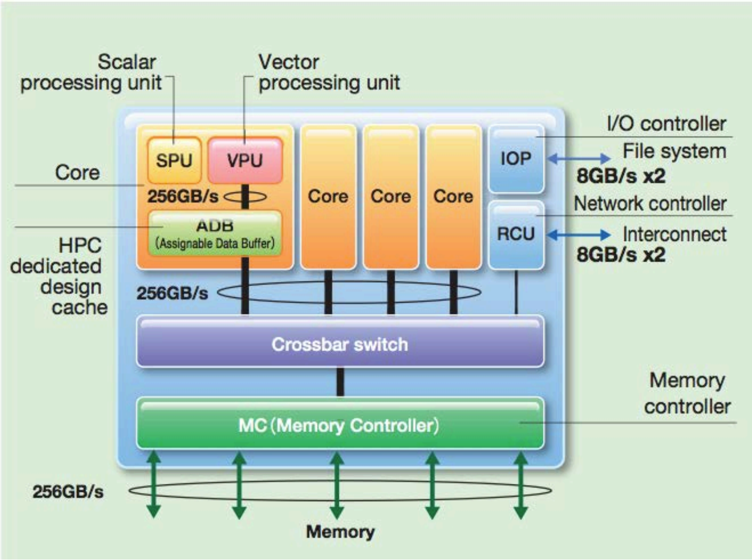
Juqueen - intra node



Juqueen - inter node

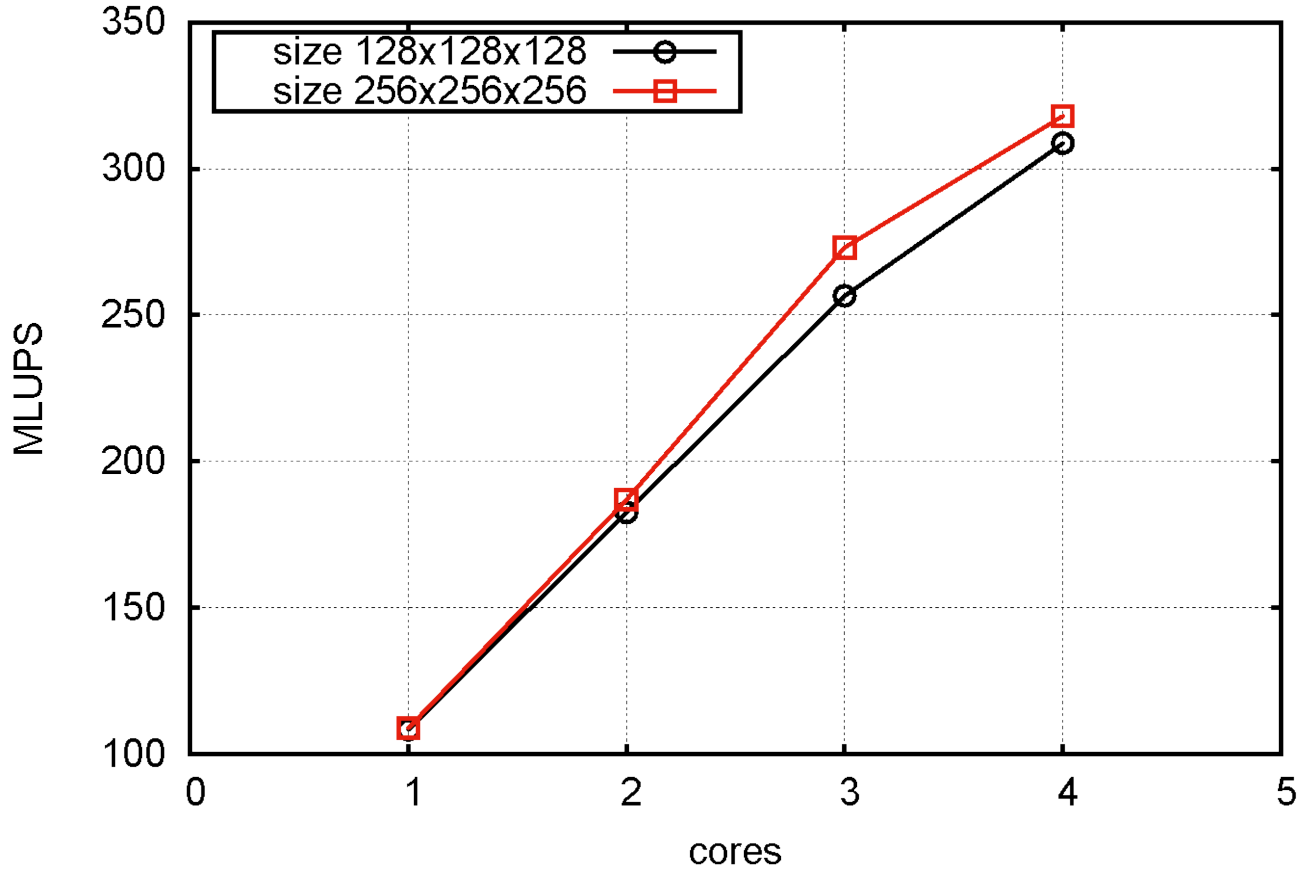


Kabuki @HLRS

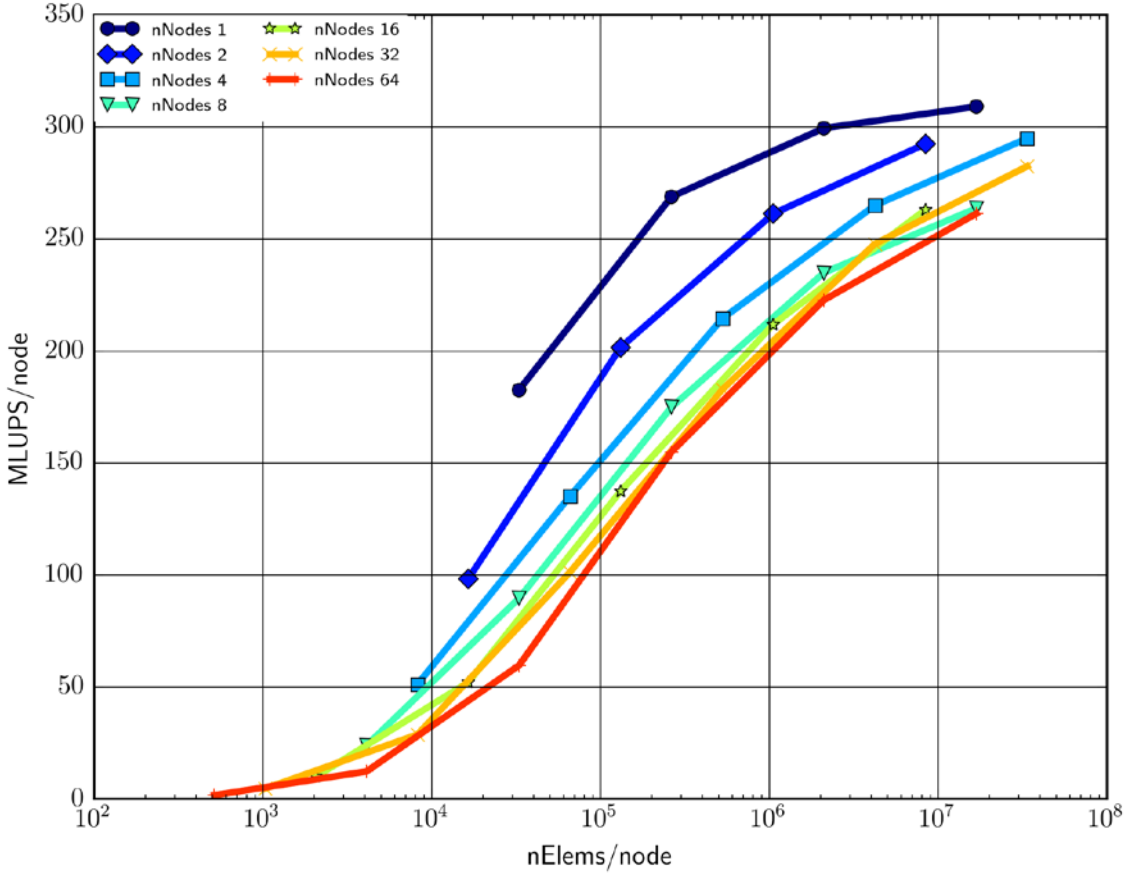


- NEC SX-ACE 4 cores CPU
  - Peak performance: 256 GFLOPS
  - max mem bandwidth: 256 GB/s
  - Memory capacity: 64 GB
  - Up to 64 nodes
- Sxf90, sxf03 Fortran compiler with mpi wraper
- Sxf03 has more restriction than gcc or intel
- Diagnostic message is especially useful for code vectorization.

Kabuki - intra node



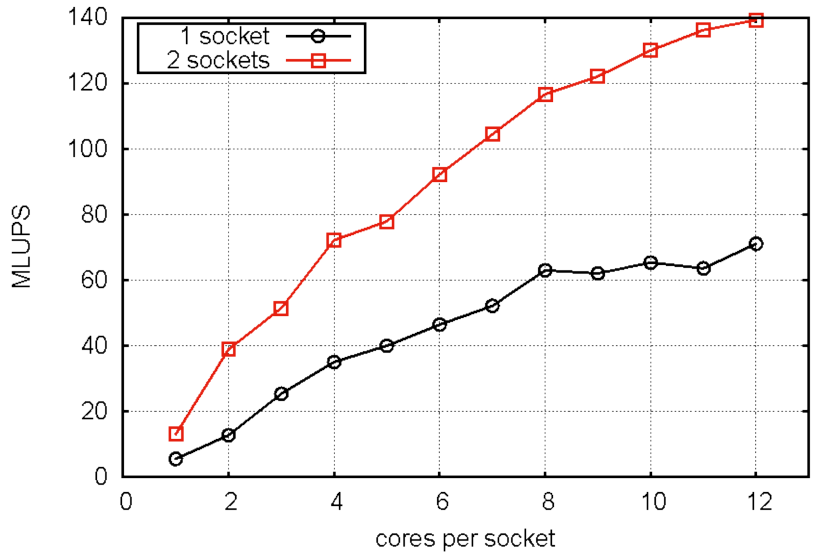
Kabuki - inter node



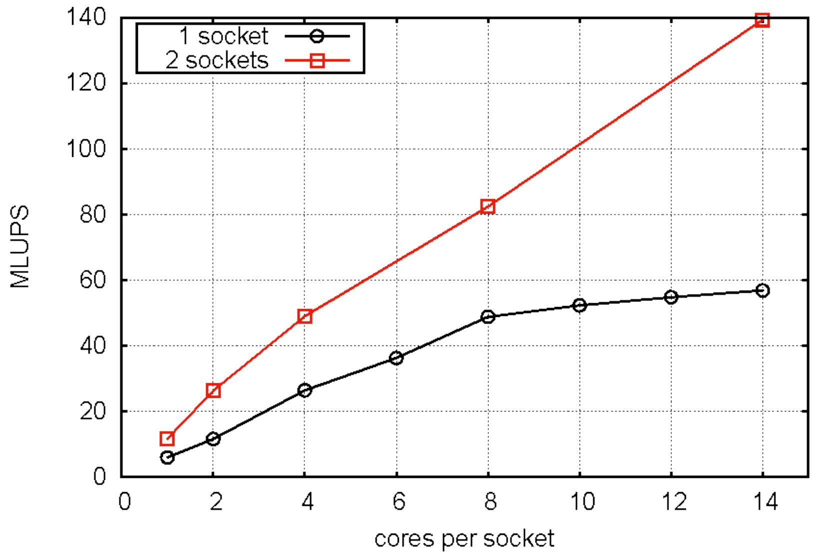
- Hornet
- Intel Haswell 12 cores
- 2 sockets and 128 GB
- Cray dragonfly topology
- intel 15 Fortran
- !DIR$ NOVECTOR

- SuperMUC phase 2
- Intel Haswell 14 cores
- 2 sockets and 64 GB
- infiniband
  - non-blocking tree
  - pruned tree
- intel 15 Fortran
- !DIR$ NOVECTOR

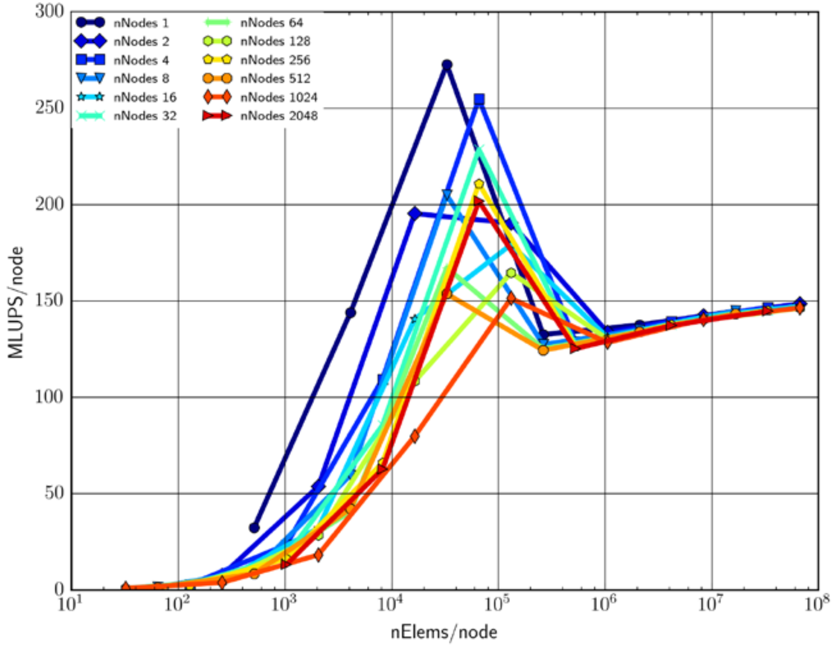
Hornet



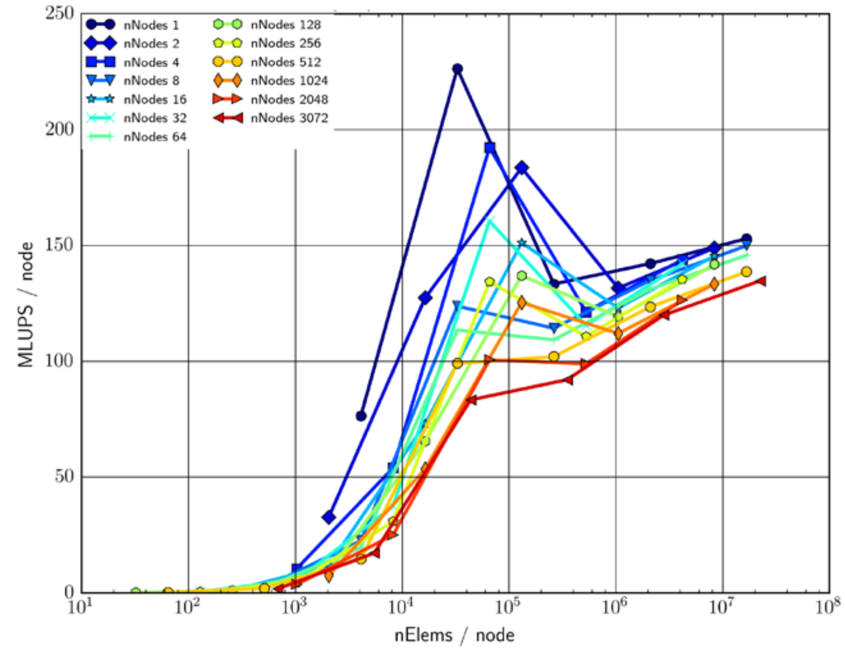
SuperMUC



Hornet



SuperMUC



Summary

- Musubi can efficiently make use of the computing resources on four different HPC systems.
  - single implementation, thus little porting efforts
  - NEC SX-ACE achieves the best single node performance.
- To further improve performance (memory bandwidth):
  - structure of array with SIMD instruction set
  - other streaming technique that can require only one set of state variables.